

IMPLEMENTATION OF DUAL-PRECISION FLOATING POINT MULTIPLIER ON FPGA

GEETHA INTI & PUSHPA KOTIPALLI

Department of Electronics and Communications, Sri Vishnu Engineering College for Women,
Bhimavaram, Andhra Pradesh, India

ABSTRACT

FPGA's have a generic nature of programmability and suppleness to implement during a wide selection of applications. However FPGA's encompasses a non-specific nature towards the scientific applications whatever floating-point operations are needed. Floating purpose arithmetic operations consume great deal of space and its resources. Typically floating purpose operations involve addition, subtraction, multiplication, division and square root. During this paper, we tend to target quick multiplication that's capable to perform either single precision operation or a double precision operation. These operations are designed to reduce the delay and space by reducing the amount of partial product generations. In this paper we tend to planned a way of Vedic multiplier method. This method shows the high performance of multiplier in embedded cores.

KEYWORDS: Partial Product Array, Vedic Multiplication, Single Precision, Doubles Precision, Verilog

INTRODUCTION

Multipliers play a dominant role in digital signal processing system (DSP). Samples of their use in implementation of algorithmic and transverse filters, discrete Fourier transforms, correlation, various measurements etc. The performance issue is powerfully depends within the effectiveness of hardware that is employed for computing multiplications. These are massively employed in application fields like multimedia systems, 3D graphics and image processing as a result of the rising and advanced technologies have allowed the numerous researchers to design multipliers which will provide each high-speed and regularity of layout, thereby creating them for suitable VLSI implementations.

In any multiplication algorithm, the operation is rotten in to partial product summation. Every pp (partial product) represents a multiple of a number to be further to the ultimate result. The majority of high-speed multipliers apply Radix-4 and Radix-8 multiplication algorithms. Between these Radix-8 encryption makes a good use of leading to less range of transistors and reduced power dissipation. However our multiplier is intended for a particular purpose and is changed the technique that is booth multiplier where the quantity of partial products is reduced to half. An ancient technique of sacred text multiplier reduces the partial product array to at least one row. This technique is very economical in terms of space and delay.

Floating-Point Number System

The term floating-point refers to the very fact that the radix point(either a decimal point or binary point)can float ,that it is placed anywhere relative to the many digits of the amount. Over the years, a spread of floating point representations is used. However, the foremost common illustration is outlined by the IEEE-754 standard. This format consists of 3 fields –a sign bit(s), a biased exponent and a mantissa (f).

There are three kinds of floating point arithmetic specifically 1) single-precision 2) double precision 3) quadruple precision. The accuracy and reliability will increase as we have tendency to move from single to quadruple precision. The accuracy strictly depends on their inputs from single precision to higher precision.

Out of which we have used single precision and double precision floating point arithmetic for multiplication. Single precision numbers have 1 bit sign bit, 8 bit exponent and 23 bit mantissa as shown in figure a. Single precision can represent 32 bits whereas double precision represents 64 bits. Double precision numbers have 1 bit sign, 11 bit exponent and 52 bit mantissa as shown in figure b.

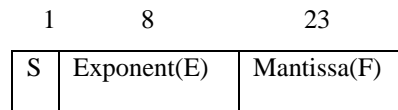


Figure A) Single-Precision Floating Point Number

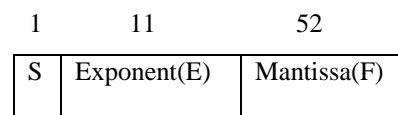


Figure B): Double-Precision Floating Point Number

Figure 1: IEEE Floating Point Formats

The 3 fields conjure a floating point number the single precision and double precision numbers are often represented by the equation 1) & 2) respectively.

$$X = (-1)^S \cdot 1.F \cdot 2^{(E-127)} \quad (1)$$

$$X = (-1)^S \cdot 1.F \cdot 2^{(E-1023)} \quad (2)$$

Where

S-sign bit

F-mantissa

E-exponent

Floating point numbers have an advantage over the fixed point is able to cover large dynamic range. The only disadvantage is that standard floating point number computations are complex to implement on hardware over fixed point calculations.

MULTIPLICATION PROCESS

Multiplying two numbers in floating point format is completed by adding the exponent of the two numbers then subtracting the bias from their result, multiplying the significand of the two numbers and calculating the sign by xoring the two sign bits. Afterwards normalizing the tentated result then rounding final result. It can be given in an algorithmic form.

Floating Point Multiplication Algorithm

- Multiply the significands i.e. (1.M1 * 1.M2)
- Placing the decimal point in the result.
- Adding the exponent i.e. (E1 + E2 - bias).

- Obtaining the sign, $s_1 \text{ xor } s_2$.
- Normalizing the result
- Rounding of the result to fit in an available bit.

This can be shown in figure 2 below

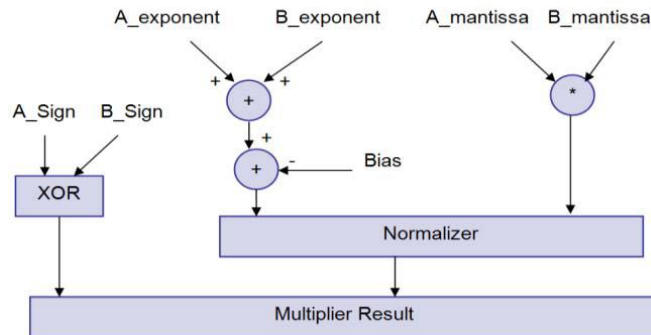


Figure 2: Floating Point Multiplier Block Diagram

HARDWARE FLOATING POINT REPRESENTATION

Sign-Bit Calculation

In the given two numbers, xor operation is performed on the two bits of sign.

Unsigned Adder (For Exponent Addition)

The operation is done on 8 bits. The unsigned adder is accountable for the addition of input of one exponent to the input of second exponent then subtracting the bias (127 or 1023). Most of the calculation time is spent on the multiplication of multiplicands, therefore a fast and quick multiplier, an 8-bit ripple carry adder is employed to add the two exponents as shown in figure 3. It is a series of cascaded full adders and one half adder. Every full adder has 3 inputs (A, B, C_i) and 2 outputs (S, C_o). The carry out (C_o) of every adder is fed to the subsequent full adder.

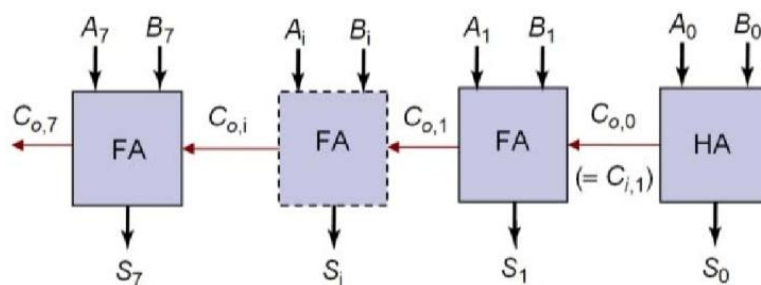


Figure 3: Ripple Carry Adder

The addition process produces a 8-bit sum (s_7 to s_0) and carry bit (c_0-7). These bits are concatenated to make a 9 bit addition result (s_8 to s_0) from that the bias is subtracted. This is often done by the array of ripple borrow subtractors. Figure 4 shows the bias subtractor that could be a chain of 7 one subtractors (os) followed by a pair of zero subtractor (zs). The borrow output of every subtractor is fed to successive subtractors. If an underflow occurs then $E_{\text{result}} < 0$. During this case output is signaled to zero.

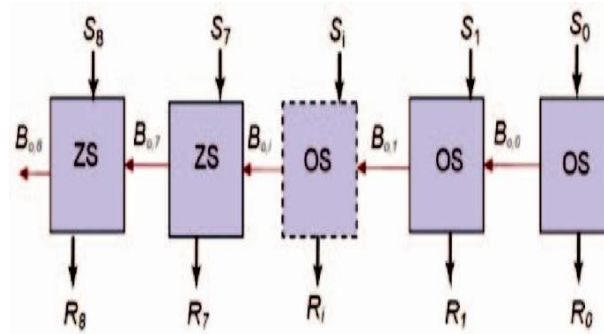


Figure 4: Ripple Borrow Sub Tractor

Multiplication of Mantissa (Vedic Multiplier)

Multiplication is based on an algorithm called *urdhva triyakbhyam* (vertical crosswise) of ancient Indian mathematics. The Sanskrit term suggest that vertical and crosswise, the thought here relies on an inspiration which ends within the generation of all partial products in conjunction with the synchronic addition of those partial products in parallel. The similarity in generation of partial product and their summation is obtained using *urdhva triyakbhyam* explained in figure 5. since there is a parallel generation of the partial product and their sums, the processor becomes independent of the clock frequency. Thus the multiplier would require constant quantity of time to calculate the product and thence is the independent of clock frequency. The advantage here is that similarity reduces the necessity of processor to work at progressively high clock frequencies. A higher clock frequency can end in raised process power, and its demerits is that it can result in raised power dissipation leading to higher device operative temperatures. By using Vedic multiplier, all the demerits related to the rise in power dissipation can be negotiated. Since its quiet faster and efficient layout features a quiet regular structure. Due to its regular structure, its layout can be done simply on a semiconductor. The Vedic multiplier has the advantage that as the number of bits increases, gate delay and area increases terribly slow as compared to other multipliers, thereby creating it time. This design is sort of economical in terms of semiconducting material area/speed.

To illustrate this multiplication theme, allow us to contemplate the multiplication of two decimal numbers. Line diagram for the multiplication is shown in figure 5. Initially the LSB digits on the both sides of the line are multiplied and added with the carry from the previous step. This generates one of the bits of the result and a carry. This carry is added in the next step and the process goes on likewise. If more than one line are there in one step, all the results are added to the previous carry. In each step, least significant bit act as the result digit and all other digits act as carry for the next step. Initially the carry is taken to be zero.

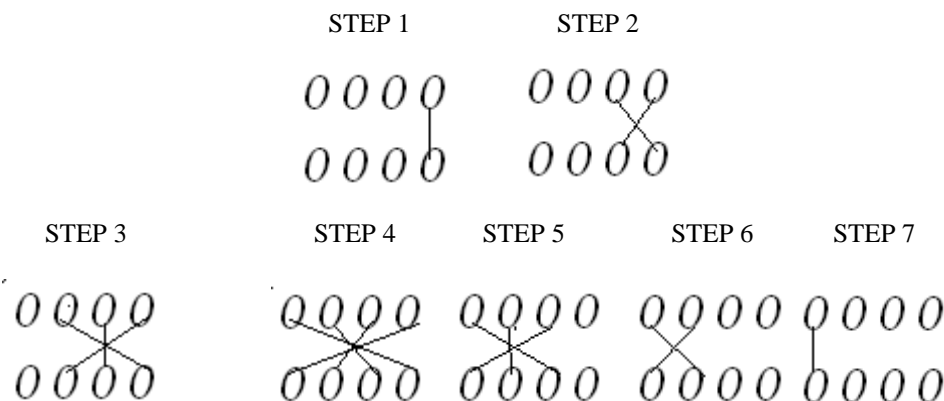


Figure 5: Line Diagram for the Vedic Multiplication

RESULTS

Single Precision Multiplier

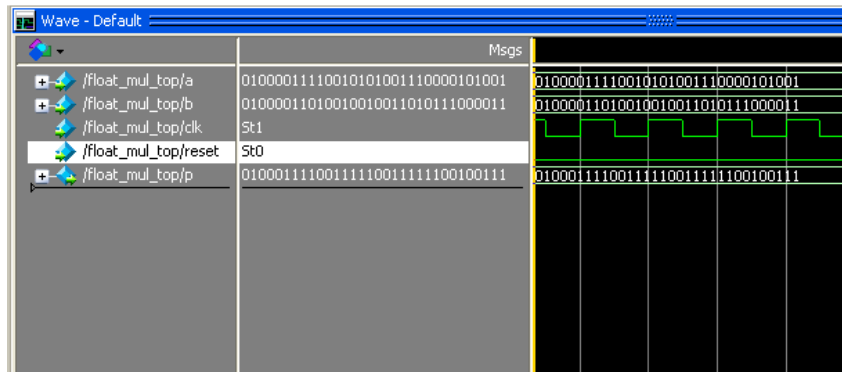


Figure 6: Simulation Results for the Proposed Vedic Multiplier

Synthesis Report for Single Precision Vedic Multiplier

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	619	4656	13%
Number of 4 input LUTs	1076	9312	11%
Number of bonded IOBs	92	232	39%

Figure 7

The simulation results for the single precision proposed method has been given in figure 6 and its synthesis is given in the table. It is clear from its report that its time consumption is typically reduced and its no of lut and its utilization of iob s are reduced when compared with single precision booth multiplier as shown in Figure 8 below.

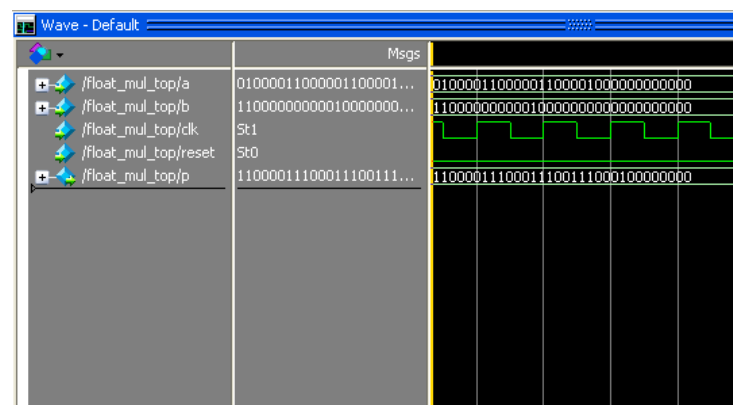


Figure 8: Simulation Results for Single Precision Booth Multiplier

Synthesis Report for Single Precision Booth Multiplier

Device Utilization Summary (estimated values)			
Logic Utilization	Used	Available	Utilization
Number of Slices	1408	4656	30%
Number of 4 input LUTs	2572	9312	27%
Number of bonded IOBs	97	232	41%

Figure 9

Single Precision Comparison Table

Table 1

Delay	Booth Multiplier	Vedic Multiplier
	82.9ns	32.7ns

Double Precision Multiplier

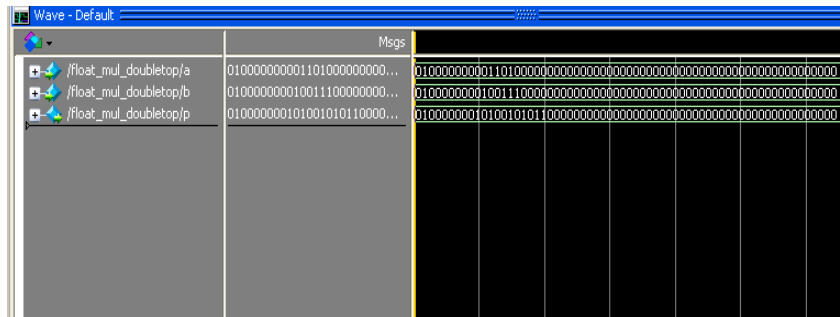


Figure 10: Simulation Results for Double Precision Booth Multiplier

Synthesis Report for Double Precision Booth Multiplier

Table 2

Device Utilization Summary			
Logic Utilization	Used	Available	Utilization
Number of 4 input LUTs	8,138	9,312	87%
Number of occupied Slices	4,654	4,656	99%
Number of Slices containing only related logic	4,654	4,654	100%
Number of Slices containing unrelated logic	0	4,654	0%
Total Number of 4 input LUTs	8,166	9,312	87%
Number used as logic	8,138		
Number used as a route-thru	28		
Number of bonded IOBs	192	190	101%
Average Fan-out of Non-Clock Nets	3.61		

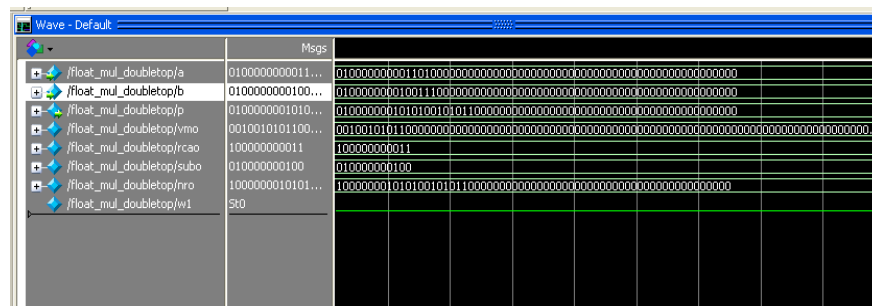


Figure 11: Simulation Results for Double Precision Vedic Multiplier

Synthesis Report for Double Precision Vedic Multiplier

Table 3

Device Utilization Summary (Estimated Values)			[-]
Logic Utilization	Used	Available	Utilization
Number of Slices	3135	4656	67%
Number of 4 input LUTs	5749	9312	61%
Number of bonded IOBs	192	190	101%

Double Precision Comparison Table

Table 4

Delay	Booth Multiplier	Vedic Multiplier
	177.198ns	145.643ns

CONCLUSIONS AND FUTURE WORK

Delay reduction is that the main concept that decides the speed of any design. The lesser the delay the economical is that the design. However reduction of delay could increase the frequency of operation. Hence there ought to be a trade-off between these two factors. The comparison table of both single and double precision multipliers represents that proposed method is more efficient than the booth multiplier. There by decreases the area. This paper presents an implementation of a floating point multiplier that supports the IEEE 754-standard binary interchange format for single precision and double precision multiplication. The multiplier can be extended to the quadruple precision and also addition and subtraction of precision numbers can execute.

REFERENCES

1. Yee Jern Chong, Student Member, IEEE, and Sri Parameswaran, Member, IEEE, "Configurable Multimode Embedded Floating-Point Units for FPGAs," in IEEE TRANSACTIONS ON VERY LARGE SCALE INTEGRATION (VLSI) SYSTEMS, VOL. 19, NO. 11, NOVEMBER 2011.
2. K.S.Hemmert and K.D.Underwood, "An analysis of the double precision Floating-point FFT on FPGAs," presented at the ACM Int.Symp. Field-program.Gate Arrays, Monterey, CA, and Feb.2004.
3. Akkas,, "Dual-mode quadruple precision floating-point adder," in *Proc. 9th Euromicro Conf. Digit. Syst. Des. (DSD)*, 2006, pp. 211–220.
4. Akkas, and M. J. Schulte, "A quadruple precision and dual double precision floating-point multiplier," in *Proc. Euromicro Symp. Digit. Syst. Des. (DSD)*, 2003, p. 76.
5. P. C. Diniz and G. Govindu, "Design of a field-programmable dual-precision Floating-point arithmetic unit," in *Proc. Int. Conf. Field Program. Logic Appl. (FPL)*, 2006, pp. 1–4.
6. Convey Computer Corporation, "Convey computer," Richardson, TX, 2008–2010 [Online]. Available: <http://www.conveycomputer.com>
7. M. J. Beauchamp, S. Hauck, and K. S. Hemmert, "Embedded floating point Units in FPGAs," in *Proc. IEEE Symp. Field Program. Gate Arrays (FPGA)*, 2006, pp. 12–20.

8. Poornima M, Shivaraj Kumar patil, Sridhar K P, Sanjay H,” Implentation of Multiplier using Vedic Algorithm,” International journal of Innovative Technology and Exploring Engineering(IJITEE) ISSN:2278-3075, Volume-2,Issue-6, May 2013.